На этом занятии мы с вами рассмотрим возможность обращаться с объектом класса, словно с объектом-функцией, то есть, применять к нему операцию круглые скобки (). Классы, объекты которых можно вызывать подобно функциям, носят название **функторы**.

Я начну с классического и простого примера реализации счетчика. Объявим следующий класс:

```language-cpp
class Counter {
    double start {0.0};
    double step {1.0};
    double count {start};

public:
    Counter(double start = 0.0, double step = 1.0) : start(start), step(step)
        { count = start; }

    operator double() const { return count; }
    double operator()()
    {
        double ret = count;
        count += step;
        return ret;
    }
};
```

Каждый объект этого класса будет иметь три переменные:

- start – начальное значение счетчика;
- step – шаг изменения счетчика;
- count – текущее значение счетчика.

С помощью конструктора можно задавать начальные величины start и step. А также преобразовывать объект класса к типу double (возвращается значение count) и применять к объектам операцию вызова функции ().

Давайте посмотрим, как это можно использовать:

```language-cpp
int main()
{
    Counter c1, c2(0.0, 0.5);

    double r1 = c1();
```

```
    c1();
    double r2 = c1;

    std::cout << r1 << " " << r2 << std::endl;

    return 0;
}
```

Создается два счетчика c1 и c2. Затем, объект c1 вызывается, как функция. В результате возвращается начальное значение count и увеличивается на величину step. Далее, мы просто применяем операцию () к объекту c1 и count еще раз увеличивается на step. А дальше вещественной переменной r2 присваиваем объект c1. В результате сработает операция приведения типа объекта к double, при которой возвращается текущее значение count. Те же самые действия можно выполнять и с объектом c2.

Конечно, в практике программирования вполне можно обойтись и без переопределения операции (). Но в ряде случаев она бывает крайне удобной. Например, представим, что нам нужно вычислять производные разных функций, заданных сигнатурой:

```language-cpp
double <имя функции>(double);
```

Это могут быть стандартные функции sin, cos и так далее. Решить эту задачу можно по разному, но мы с вами сделаем это с помощью следующего класса:

```language-cpp
using func_diff_dobule = double(*)(double);

class Diff {
    func_diff_dobule func {nullptr};
    double dt {0.001};

public:
    Diff(func_diff_dobule f, double delta = 0.001) : func(f), dt(delta)
        { }

    double operator()(double x)
    {
        return (func(x+dt)-func(x)) / dt;
    }
};
```

Здесь определяется тип func_diff_dobule указателя на функцию. А в самом классе Diff две переменные:

- func – указатель на функцию, для которой будут вычисляться производные;
- dt – интервал для численного получения значения производной любой непрерывной функции.

Напомню, что производную любой функции f в точке x можно найти по формуле:

```language-cpp
diff(x) = (f(x+dt) - f(x)) / dt
```

Именно это происходит при вызове операции ().

Воспользоваться классом Diff можно следующим образом:

```language-cpp
#include <iostream>
#include <math.h>

int main()
{
    Diff sin_diff(sin);

    double res = sin_diff(3.1415/2.0);
    std::cout << res << std::endl;

    return 0;
}
```

Получилось очень удобно и интуитивно понятно.